Network Working Group
Request for Comments: 4379
Updates: 1122
Category: Standards Track

K. Kompella
Juniper Networks, Inc.
G. Swallow
Cisco Systems, Inc.

February 2006

Detecting Multi-Protocol Label Switched (MPLS) Data Plane Failures

Status of This Memo

Copyright Notice

Abstract

This document describes a simple and efficient mechanism that can be used to detect data plane failures in Multi-Protocol Label Switching (MPLS) Label Switched Paths (LSPs). There are two parts to this document: information carried in an MPLS "echo request" and "echo reply" for the purposes of fault detection and isolation, and mechanisms for reliably sending the echo reply.

Table of Contents

1. Introduction

This document describes a simple and efficient mechanism that can be used to detect data plane failures in MPLS Label Switched Paths (LSPs). There are two parts to this document: information carried in an MPLS "echo request" and "echo reply", and mechanisms for transporting the echo reply. The first part aims at providing enough information to check correct operation of the data plane, as well as a mechanism to verify the data plane against the control plane, and thereby localize faults. The second part suggests two methods of reliable reply channels for the echo request message for more robust fault isolation.

An important consideration in this design is that MPLS echo requests follow the same data path that normal MPLS packets would traverse. MPLS echo requests are meant primarily to validate the data plane, and secondarily to verify the data plane against the control plane. Mechanisms to check the control plane are valuable, but are not covered in this document.

This document makes special use of the address range 127/8. This is an exception to the behavior defined in RFC 1122 [RFC1122] and updates that RFC. The motivation for this change and the details of this exceptional use are discussed in section 2.1 below.

1.1. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [KEYWORDS].

The term "Must Be Zero" (MBZ) is used in object descriptions for reserved fields. These fields MUST be set to zero when sent and ignored on receipt.

Terminology pertaining to L2 and L3 Virtual Private Networks (VPNs) is defined in [RFC4026].

Since this document refers to the MPLS Time to Live (TTL) far more frequently than the IP TTL, the authors have chosen the convention of using the unqualified "TTL" to mean "MPLS TTL" and using "IP TTL" for the TTL value in the IP header.

1.2. Structure of This Document

The body of this memo contains four main parts: motivation, MPLS echo request/reply packet format, LSP ping operation, and a reliable return path. It is suggested that first-time readers skip the actual packet formats and read the Theory of Operation first; the document is structured the way it is to avoid forward references.

1.3. Contributors

The following made vital contributions to all aspects of this document, and much of the material came out of debate and discussion among this group.

Ronald P. Bonica, Juniper Networks, Inc.
Dave Cooper, Global Crossing
Ping Pan, Hammerhead Systems

Nischal Sheth, Juniper Networks, Inc.
Sanjay Wadhwa, Juniper Networks, Inc.

2. Motivation

When an LSP fails to deliver user traffic, the failure cannot always be detected by the MPLS control plane. There is a need to provide a tool that would enable users to detect such traffic "black holes" or misrouting within a reasonable period of time, and a mechanism to isolate faults.

In this document, we describe a mechanism that accomplishes these goals. This mechanism is modeled after the ping/traceroute paradigm: ping (ICMP echo request [ICMP]) is used for connectivity checks, and traceroute is used for hop-by-hop fault localization as well as path tracing. This document specifies a "ping" mode and a "traceroute" mode for testing MPLS LSPs.

The basic idea is to verify that packets that belong to a particular Forwarding Equivalence Class (FEC) actually end their MPLS path on a Label Switching Router (LSR) that is an egress for that FEC. This document proposes that this test be carried out by sending a packet (called an "MPLS echo request") along the same data path as other packets belonging to this FEC. An MPLS echo request also carries information about the FEC whose MPLS path is being verified. This echo request is forwarded just like any other packet belonging to that FEC. In "ping" mode (basic connectivity check), the packet should reach the end of the path, at which point it is sent to the control plane of the egress LSR, which then verifies whether it is indeed an egress for the FEC. In "traceroute" mode (fault isolation), the packet is sent to the control plane of each transit LSR, which performs various checks that it is indeed a transit LSR for this path; this LSR also returns further information that helps check the control plane against the data plane, i.e., that forwarding matches what the routing protocols determined as the path.

One way these tools can be used is to periodically ping an FEC to ensure connectivity. If the ping fails, one can then initiate a traceroute to determine where the fault lies. One can also periodically traceroute FECs to verify that forwarding matches the control plane; however, this places a greater burden on transit LSRs and thus should be used with caution.

2.1. Use of Address Range 127/8

As described above, LSP ping is intended as a diagnostic tool. It is intended to enable providers of an MPLS-based service to isolate network faults. In particular, LSP ping needs to diagnose situations where the control and data planes are out of sync. It performs this by routing an MPLS echo request packet based solely on its label stack. That is, the IP destination address is never used in a forwarding decision. In fact, the sender of an MPLS echo request packet may not know, a priori, the address of the router at the end of the LSP.

Providers of MPLS-based services also need the ability to trace all of the possible paths that an LSP may take. Since most MPLS services are based on IP unicast forwarding, these paths are subject to equal-cost multi-path (ECMP) load sharing.

This leads to the following requirements:

1. Although the LSP in question may be broken in unknown ways, the likelihood of a diagnostic packet being delivered to a user of an MPLS service MUST be held to an absolute minimum.

2. If an LSP is broken in such a way that it prematurely terminates, the diagnostic packet MUST NOT be IP forwarded.

3. A means of varying the diagnostic packets such that they exercise all ECMP paths is thus REQUIRED.

Clearly, using general unicast addresses satisfies neither of the first two requirements. A number of other options for addresses were considered, including a portion of the private address space (as determined by the network operator) and the newly designated IPv4 link local addresses. Use of the private address space was deemed ineffective since the leading MPLS-based service is an IPv4 Virtual Private Network (VPN). VPNs often use private addresses.

The IPv4 link local addresses are more attractive in that the scope over which they can be forwarded is limited. However, if one were to use an address from this range, it would still be possible for the first recipient of a diagnostic packet that "escaped" from a broken LSP to have that address assigned to the interface on which it arrived and thus could mistakenly receive such a packet. Furthermore, the IPv4 link local address range has only recently been allocated. Many deployed routers would forward a packet with an address from that range toward the default route.

The 127/8 range for IPv4 and that same range embedded in as IPv4-mapped IPv6 addresses for IPv6 was chosen for a number of reasons.

RFC 1122 allocates the 127/8 as "Internal host loopback address" and states: "Addresses of this form MUST NOT appear outside a host." Thus, the default behavior of hosts is to discard such packets. This helps to ensure that if a diagnostic packet is misdirected to a host, it will be silently discarded.

RFC 1812 [RFC1812] states:

> A router SHOULD NOT forward, except over a loopback interface, any packet that has a destination address on network 127. A router MAY have a switch that allows the network manager to disable these checks. If such a switch is provided, it MUST default to performing the checks.

This helps to ensure that diagnostic packets are never IP forwarded.

The 127/8 address range provides 16M addresses allowing wide flexibility in varying addresses to exercise ECMP paths. Finally, as an implementation optimization, the 127/8 provides an easy means of identifying possible LSP packets.

3. Packet Format

An MPLS echo request is a (possibly labeled) IPv4 or IPv6 UDP packet; the contents of the UDP packet have the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Version Number        |         Global Flags          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Message Type |   Reply mode  |  Return Code  | Return Subcode|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sender’s Handle                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    TimeStamp Sent (seconds)                   |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  TimeStamp Sent (microseconds)                |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  TimeStamp Received (seconds)                 |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                TimeStamp Received (microseconds)              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            TLVs ...                           |
.                                                               .
.                                                               .
.                                                               .
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Version Number is currently 1. (Note: the version number is to be incremented whenever a change is made that affects the ability of an implementation to correctly parse or process an MPLS echo request/reply. These changes include any syntactic or semantic changes made to any of the fixed fields, or to any Type-Length-Value (TLV) or sub-TLV assignment or format that is defined at a certain version number. The version number may not need to be changed if an optional TLV or sub-TLV is added.)

The Global Flags field is a bit vector with the following format:

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             MBZ             |V|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

One flag is defined for now, the V bit; the rest MUST be set to zero when sending and ignored on receipt.

The V (Validate FEC Stack) flag is set to 1 if the sender wants the receiver to perform FEC Stack validation; if V is 0, the choice is left to the receiver.

The Message Type is one of the following:

Value	Meaning
1	MPLS echo request
2	MPLS echo reply

The Reply Mode can take one of the following values:

Value	Meaning
1	Do not reply
2	Reply via an IPv4/IPv6 UDP packet
3	Reply via an IPv4/IPv6 UDP packet with Router Alert
4	Reply via application level control channel

An MPLS echo request with 1 (Do not reply) in the Reply Mode field may be used for one-way connectivity tests; the receiving router may log gaps in the Sequence Numbers and/or maintain delay/jitter statistics. An MPLS echo request would normally have 2 (Reply via an IPv4/IPv6 UDP packet) in the Reply Mode field. If the normal IP return path is deemed unreliable, one may use 3 (Reply via an IPv4/IPv6 UDP packet with Router Alert). Note that this requires that all intermediate routers understand and know how to forward MPLS echo replies. The echo reply uses the same IP version number as the received echo request, i.e., an IPv4 encapsulated echo reply is sent in response to an IPv4 encapsulated echo request.

Some applications support an IP control channel. One such example is the associated control channel defined in Virtual Circuit Connectivity Verification (VCCV) [VCCV]. Any application that supports an IP control channel between its control entities may set the Reply Mode to 4 (Reply via application level control channel) to ensure that replies use that same channel. Further definition of this codepoint is application specific and thus beyond the scope of this document.

Return Codes and Subcodes are described in the next section.

The Sender’s Handle is filled in by the sender, and returned unchanged by the receiver in the echo reply (if any). There are no semantics associated with this handle, although a sender may find this useful for matching up requests with replies.

The Sequence Number is assigned by the sender of the MPLS echo request and can be (for example) used to detect missed replies.

The TimeStamp Sent is the time-of-day (in seconds and microseconds, according to the sender’s clock) in NTP format [NTP] when the MPLS echo request is sent. The TimeStamp Received in an echo reply is the time-of-day (according to the receiver’s clock) in NTP format that the corresponding echo request was received.

TLVs (Type-Length-Value tuples) have the following format:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |             Type              |            Length             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                             Value                             |
   .                                                               .
   .                                                               .
   .                                                               .
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Types are defined below; Length is the length of the Value field in octets. The Value field depends on the Type; it is zero padded to align to a 4-octet boundary. TLVs may be nested within other TLVs, in which case the nested TLVs are called sub-TLVs. Sub-TLVs have independent types and MUST also be 4-octet aligned.

Two examples follow. The Label Distribution Protocol (LDP) IPv4 FEC sub-TLV has the following format:

```
       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |    Type = 1 (LDP IPv4 FEC)    |          Length = 5           |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                          IPv4 prefix                          |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      | Prefix Length |         Must Be Zero                          |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Length for this TLV is 5. A Target FEC Stack TLV that contains an LDP IPv4 FEC sub-TLV and a VPN IPv4 prefix sub-TLV has the following format:

```
       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |      Type = 1 (FEC TLV)       |          Length = 12          |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |  sub-Type = 1 (LDP IPv4 FEC)  |          Length = 5           |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                          IPv4 prefix                          |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      | Prefix Length |         Must Be Zero                          |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      | sub-Type = 6 (VPN IPv4 prefix)|          Length = 13          |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                      Route Distinguisher                      |
      |                          (8 octets)                           |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                         IPv4 prefix                           |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      | Prefix Length |                 Must Be Zero                  |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

A description of the Types and Values of the top-level TLVs for LSP ping are given below:

Type #	Value Field
1	Target FEC Stack
2	Downstream Mapping
3	Pad
4	Not Assigned
5	Vendor Enterprise Number
6	Not Assigned
7	Interface and Label Stack
8	Not Assigned
9	Errored TLVs
10	Reply TOS Byte

Types less than 32768 (i.e., with the high-order bit equal to 0) are mandatory TLVs that MUST either be supported by an implementation or result in the return code of 2 ("One or more of the TLVs was not understood") being sent in the echo response.

Types greater than or equal to 32768 (i.e., with the high-order bit equal to 1) are optional TLVs that SHOULD be ignored if the implementation does not understand or support them.

3.1. Return Codes

The Return Code is set to zero by the sender. The receiver can set it to one of the values listed below. The notation <RSC> refers to the Return Subcode. This field is filled in with the stack-depth for those codes that specify that. For all other codes, the Return Subcode MUST be set to zero.

Value	Meaning
0	No return code
1	Malformed echo request received
2	One or more of the TLVs was not understood
3	Replying router is an egress for the FEC at stack-depth <RSC>
4	Replying router has no mapping for the FEC at stack-depth <RSC>

5 Downstream Mapping Mismatch (See Note 1)

6 Upstream Interface Index Unknown (See Note 1)

7 Reserved

8 Label switched at stack-depth <RSC>

9 Label switched but no MPLS forwarding at stack-depth <RSC>

10 Mapping for this FEC is not the given label at stack-depth <RSC>

11 No label entry at stack-depth <RSC>

12 Protocol not associated with interface at FEC stack-depth <RSC>

13 Premature termination of ping due to label stack shrinking to a single label

Note 1

The Return Subcode contains the point in the label stack where processing was terminated. If the RSC is 0, no labels were processed. Otherwise the packet would have been label switched at depth RSC.

3.2. Target FEC Stack

A Target FEC Stack is a list of sub-TLVs. The number of elements is determined by looking at the sub-TLV length fields.

Sub-Type	Length	Value Field
1	5	LDP IPv4 prefix
2	17	LDP IPv6 prefix
3	20	RSVP IPv4 LSP
4	56	RSVP IPv6 LSP
5		Not Assigned
6	13	VPN IPv4 prefix
7	25	VPN IPv6 prefix
8	14	L2 VPN endpoint
9	10	"FEC 128" Pseudowire (deprecated)
10	14	"FEC 128" Pseudowire
11	16+	"FEC 129" Pseudowire
12	5	BGP labeled IPv4 prefix

13	17	BGP labeled IPv6 prefix
14	5	Generic IPv4 prefix
15	17	Generic IPv6 prefix
16	4	Nil FEC

Other FEC Types will be defined as needed.

Note that this TLV defines a stack of FECs, the first FEC element corresponding to the top of the label stack, etc.

An MPLS echo request MUST have a Target FEC Stack that describes the FEC Stack being tested. For example, if an LSR X has an LDP mapping [LDP] for 192.168.1.1 (say, label 1001), then to verify that label 1001 does indeed reach an egress LSR that announced this prefix via LDP, X can send an MPLS echo request with an FEC Stack TLV with one FEC in it, namely, of type LDP IPv4 prefix, with prefix 192.168.1.1/32, and send the echo request with a label of 1001.

Say LSR X wanted to verify that a label stack of <1001, 23456> is the right label stack to use to reach a VPN IPv4 prefix [see section 3.2.5] of 10/8 in VPN foo. Say further that LSR Y with loopback address 192.168.1.1 announced prefix 10/8 with Route Distinguisher RD-foo-Y (which may in general be different from the Route Distinguisher that LSR X uses in its own advertisements for VPN foo), label 23456 and BGP next hop 192.168.1.1 [BGP]. Finally, suppose that LSR X receives a label binding of 1001 for 192.168.1.1 via LDP. X has two choices in sending an MPLS echo request: X can send an MPLS echo request with an FEC Stack TLV with a single FEC of type VPN IPv4 prefix with a prefix of 10/8 and a Route Distinguisher of RD-foo-Y. Alternatively, X can send an FEC Stack TLV with two FECs, the first of type LDP IPv4 with a prefix of 192.168.1.1/32 and the second of type of IP VPN with a prefix 10/8 with Route Distinguisher of RD-foo-Y. In either case, the MPLS echo request would have a label stack of <1001, 23456>. (Note: in this example, 1001 is the "outer" label and 23456 is the "inner" label.)

3.2.1. LDP IPv4 Prefix

The IPv4 Prefix FEC is defined in [LDP]. When an LDP IPv4 prefix is encoded in a label stack, the following format is used. The value consists of 4 octets of an IPv4 prefix followed by 1 octet of prefix length in bits; the format is given below. The IPv4 prefix is in network byte order; if the prefix is shorter than 32 bits, trailing bits SHOULD be set to zero. See [LDP] for an example of a Mapping for an IPv4 FEC.

```
       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                          IPv4 prefix                          |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      | Prefix Length |         Must Be Zero                          |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.2.2. LDP IPv6 Prefix

The IPv6 Prefix FEC is defined in [LDP]. When an LDP IPv6 prefix is encoded in a label stack, the following format is used. The value consists of 16 octets of an IPv6 prefix followed by 1 octet of prefix length in bits; the format is given below. The IPv6 prefix is in network byte order; if the prefix is shorter than 128 bits, the trailing bits SHOULD be set to zero. See [LDP] for an example of a Mapping for an IPv6 FEC.

```
       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                          IPv6 prefix                          |
      |                          (16 octets)                          |
      |                                                               |
      |                                                               |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      | Prefix Length |         Must Be Zero                          |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.2.3. RSVP IPv4 LSP

The value has the format below. The value fields are taken from RFC 3209, sections 4.6.1.1 and 4.6.2.1. See [RSVP-TE].

```
       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                 IPv4 tunnel end point address                 |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |          Must Be Zero         |     Tunnel ID                 |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                       Extended Tunnel ID                      |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                   IPv4 tunnel sender address                  |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |          Must Be Zero         |            LSP ID             |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.2.4. RSVP IPv6 LSP

The value has the format below. The value fields are taken from RFC 3209, sections 4.6.1.2 and 4.6.2.2. See [RSVP-TE].

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                  IPv6 tunnel end point address                |
   |                                                               |
   |                                                               |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          Must Be Zero         |          Tunnel ID            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       Extended Tunnel ID                      |
   |                                                               |
   |                                                               |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                   IPv6 tunnel sender address                  |
   |                                                               |
   |                                                               |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |          Must Be Zero         |            LSP ID             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.2.5. VPN IPv4 Prefix

VPN-IPv4 Network Layer Routing Information (NLRI) is defined in [RFC4365]. This document uses the term VPN IPv4 prefix for a VPN-IPv4 NLRI that has been advertised with an MPLS label in BGP. See [BGP-LABEL].

When a VPN IPv4 prefix is encoded in a label stack, the following format is used. The value field consists of the Route Distinguisher advertised with the VPN IPv4 prefix, the IPv4 prefix (with trailing 0 bits to make 32 bits in all), and a prefix length, as follows:

```
       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                      Route Distinguisher                      |
      |                          (8 octets)                           |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                         IPv4 prefix                           |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      | Prefix Length |                 Must Be Zero                  |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Route Distinguisher (RD) is an 8-octet identifier; it does not contain any inherent information. The purpose of the RD is solely to allow one to create distinct routes to a common IPv4 address prefix. The encoding of the RD is not important here. When matching this field to the local FEC information, it is treated as an opaque value.

3.2.6. VPN IPv6 Prefix

VPN-IPv6 Network Layer Routing Information (NLRI) is defined in [RFC4365]. This document uses the term VPN IPv6 prefix for a VPN-IPv6 NLRI that has been advertised with an MPLS label in BGP. See [BGP-LABEL].

When a VPN IPv6 prefix is encoded in a label stack, the following format is used. The value field consists of the Route Distinguisher advertised with the VPN IPv6 prefix, the IPv6 prefix (with trailing 0 bits to make 128 bits in all), and a prefix length, as follows:

```
       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                      Route Distinguisher                      |
      |                          (8 octets)                           |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                         IPv6 prefix                           |
      |                                                               |
      |                                                               |
      |                                                               |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      | Prefix Length |                 Must Be Zero                  |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The Route Distinguisher is identical to the VPN IPv4 Prefix RD, except that it functions here to allow the creation of distinct routes to IPv6 prefixes. See section 3.2.5. When matching this field to local FEC information, it is treated as an opaque value.

3.2.7. L2 VPN Endpoint

VPLS stands for Virtual Private LAN Service. The terms VPLS BGP NLRI and VE ID (VPLS Edge Identifier) are defined in [VPLS-BGP]. This document uses the simpler term L2 VPN endpoint when referring to a VPLS BGP NLRI. The Route Distinguisher is an 8-octet identifier used to distinguish information about various L2 VPNs advertised by a node. The VE ID is a 2-octet identifier used to identify a particular node that serves as the service attachment point within a VPLS. The structure of these two identifiers is unimportant here; when matching these fields to local FEC information, they are treated as opaque values. The encapsulation type is identical to the PW Type in section 3.2.8 below.

When an L2 VPN endpoint is encoded in a label stack, the following format is used. The value field consists of a Route Distinguisher (8 octets), the sender (of the ping)’s VE ID (2 octets), the receiver’s VE ID (2 octets), and an encapsulation type (2 octets), formatted as follows:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      Route Distinguisher                      |
   |                          (8 octets)                           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |         Sender’s VE ID        |       Receiver’s VE ID        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      Encapsulation Type       |         Must Be Zero          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.2.8. FEC 128 Pseudowire (Deprecated)

FEC 128 (0x80) is defined in [PW-CONTROL], as are the terms PW ID (Pseudowire ID) and PW Type (Pseudowire Type). A PW ID is a non-zero 32-bit connection ID. The PW Type is a 15-bit number indicating the encapsulation type. It is carried right justified in the field below termed encapsulation type with the high-order bit set to zero. Both of these fields are treated in this protocol as opaque values.

When an FEC 128 is encoded in a label stack, the following format is used. The value field consists of the remote PE address (the destination address of the targeted LDP session), the PW ID, and the encapsulation type as follows:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      Remote PE Address                        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                             PW ID                             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |            PW Type            |          Must Be Zero         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

This FEC is deprecated and is retained only for backward compatibility. Implementations of LSP ping SHOULD accept and process this TLV, but SHOULD send LSP ping echo requests with the new TLV (see next section), unless explicitly configured to use the old TLV.

An LSR receiving this TLV SHOULD use the source IP address of the LSP echo request to infer the sender’s PE address.

3.2.9. FEC 128 Pseudowire (Current)

FEC 128 (0x80) is defined in [PW-CONTROL], as are the terms PW ID (Pseudowire ID) and PW Type (Pseudowire Type). A PW ID is a non-zero 32-bit connection ID. The PW Type is a 15-bit number indicating the encapsulation type. It is carried right justified in the field below termed encapsulation type with the high-order bit set to zero.

Both of these fields are treated in this protocol as opaque values. When matching these field to the local FEC information, the match MUST be exact.

When an FEC 128 is encoded in a label stack, the following format is used. The value field consists of the sender’s PE address (the source address of the targeted LDP session), the remote PE address (the destination address of the targeted LDP session), the PW ID, and the encapsulation type as follows:

```
       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                     Sender’s PE Address                       |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                      Remote PE Address                        |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                             PW ID                             |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |            PW Type            |          Must Be Zero         |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.2.10. FEC 129 Pseudowire

FEC 129 (0x81) and the terms PW Type, Attachment Group Identifier (AGI), Attachment Group Identifier Type (AGI Type), Attachment Individual Identifier Type (AII Type), Source Attachment Individual Identifier (SAII), and Target Attachment Individual Identifier (TAII) are defined in [PW-CONTROL]. The PW Type is a 15-bit number indicating the encapsulation type. It is carried right justified in the field below PW Type with the high-order bit set to zero. All the other fields are treated as opaque values and copied directly from the FEC 129 format. All of these values together uniquely define the FEC within the scope of the LDP session identified by the source and remote PE addresses.

When an FEC 129 is encoded in a label stack, the following format is used. The Length of this TLV is 16 + AGI length + SAII length + TAII length. Padding is used to make the total length a multiple of 4; the length of the padding is not included in the Length field.

```
       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                     Sender’s PE Address                       |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                      Remote PE Address                        |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |            PW Type            |   AGI Type    |  AGI Length   |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      ~                           AGI Value                           ~
      |                                                               |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |   AII Type    |  SAII Length  |      SAII Value               |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      ~                    SAII Value (continued)                     ~
      |                                                               |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |   AII Type    |  TAII Length  |      TAII Value               |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      ~                    TAII Value (continued)                     ~
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |  TAII (cont.) |  0-3 octets of zero padding                   |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.2.11. BGP Labeled IPv4 Prefix

BGP labeled IPv4 prefixes are defined in [BGP-LABEL]. When a BGP labeled IPv4 prefix is encoded in a label stack, the following format is used. The value field consists the IPv4 prefix (with trailing 0 bits to make 32 bits in all), and the prefix length, as follows:

```
       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                          IPv4 Prefix                          |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      | Prefix Length |                 Must Be Zero                  |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.2.12. BGP Labeled IPv6 Prefix

BGP labeled IPv6 prefixes are defined in [BGP-LABEL]. When a BGP labeled IPv6 prefix is encoded in a label stack, the following format is used. The value consists of 16 octets of an IPv6 prefix followed by 1 octet of prefix length in bits; the format is given below. The IPv6 prefix is in network byte order; if the prefix is shorter than 128 bits, the trailing bits SHOULD be set to zero.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          IPv6 prefix                          |
   |                          (16 octets)                          |
   |                                                               |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Prefix Length |         Must Be Zero                          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.2.13. Generic IPv4 Prefix

The value consists of 4 octets of an IPv4 prefix followed by 1 octet of prefix length in bits; the format is given below. The IPv4 prefix is in network byte order; if the prefix is shorter than 32 bits, trailing bits SHOULD be set to zero. This FEC is used if the protocol advertising the label is unknown or may change during the course of the LSP. An example is an inter-AS LSP that may be signaled by LDP in one Autonomous System (AS), by RSVP-TE [RSVP-TE] in another AS, and by BGP between the ASes, such as is common for inter-AS VPNs.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          IPv4 prefix                          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Prefix Length |         Must Be Zero                          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.2.14. Generic IPv6 Prefix

The value consists of 16 octets of an IPv6 prefix followed by 1 octet of prefix length in bits; the format is given below. The IPv6 prefix is in network byte order; if the prefix is shorter than 128 bits, the trailing bits SHOULD be set to zero.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                          IPv6 prefix                          |
   |                          (16 octets)                          |
   |                                                               |
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Prefix Length |         Must Be Zero                          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.2.15. Nil FEC

At times, labels from the reserved range, e.g., Router Alert and Explicit-null, may be added to the label stack for various diagnostic purposes such as influencing load-balancing. These labels may have no explicit FEC associated with them. The Nil FEC Stack is defined to allow a Target FEC Stack sub-TLV to be added to the Target FEC Stack to account for such labels so that proper validation can still be performed.

The Length is 4. Labels are 20-bit values treated as numbers.

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                 Label                 |          MBZ          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Label is the actual label value inserted in the label stack; the MBZ fields MUST be zero when sent and ignored on receipt.

3.3. Downstream Mapping

The Downstream Mapping object is a TLV that MAY be included in an echo request message. Only one Downstream Mapping object may appear in an echo request. The presence of a Downstream Mapping object is a request that Downstream Mapping objects be included in the echo reply. If the replying router is the destination of the FEC, then a Downstream Mapping TLV SHOULD NOT be included in the echo reply. Otherwise the replying router SHOULD include a Downstream Mapping object for each interface over which this FEC could be forwarded. For a more precise definition of the notion of "downstream", see section 3.3.2, "Downstream Router and Interface".

The Length is K + M + 4*N octets, where M is the Multipath Length, and N is the number of Downstream Labels. Values for K are found in the description of Address Type below. The Value field of a Downstream Mapping has the following format:

```
       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |               MTU             | Address Type  |    DS Flags   |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |             Downstream IP Address (4 or 16 octets)            |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |         Downstream Interface Address (4 or 16 octets)         |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      | Multipath Type| Depth Limit   |        Multipath Length       |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      .                                                               .
      .                     (Multipath Information)                   .
      .                                                               .
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |               Downstream Label                |    Protocol   |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      .                                                               .
      .                                                               .
      .                                                               .
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |               Downstream Label                |    Protocol   |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Maximum Transmission Unit (MTU)

The MTU is the size in octets of the largest MPLS frame (including label stack) that fits on the interface to the Downstream LSR.

Address Type

The Address Type indicates if the interface is numbered or unnumbered. It also determines the length of the Downstream IP Address and Downstream Interface fields. The resulting total for the initial part of the TLV is listed in the table below as "K Octets". The Address Type is set to one of the following values:

Type #	Address Type	K Octets
1	IPv4 Numbered	16
2	IPv4 Unnumbered	16
3	IPv6 Numbered	40
4	IPv6 Unnumbered	28

DS Flags

The DS Flags field is a bit vector with the following format:

```
 0 1 2 3 4 5 6 7
+-+-+-+-+-+-+-+-+
| Rsvd(MBZ) |I|N|
+-+-+-+-+-+-+-+-+
```

Two flags are defined currently, I and N. The remaining flags MUST be set to zero when sending and ignored on receipt.

Flag Name and Meaning
---- ----------------

I Interface and Label Stack Object Request

When this flag is set, it indicates that the replying router SHOULD include an Interface and Label Stack Object in the echo reply message.

N Treat as a Non-IP Packet

Echo request messages will be used to diagnose non-IP flows. However, these messages are carried in IP packets. For a router that alters its ECMP algorithm based on the FEC or deep packet examination, this flag requests that the router treat this as it would if the determination of an IP payload had failed.

Downstream IP Address and Downstream Interface Address

IPv4 addresses and interface indices are encoded in 4 octets; IPv6 addresses are encoded in 16 octets.

If the interface to the downstream LSR is numbered, then the Address Type MUST be set to IPv4 or IPv6, the Downstream IP Address MUST be set to either the downstream LSR’s Router ID or the interface address of the downstream LSR, and the Downstream Interface Address MUST be set to the downstream LSR’s interface address.

If the interface to the downstream LSR is unnumbered, the Address Type MUST be IPv4 Unnumbered or IPv6 Unnumbered, the Downstream IP Address MUST be the downstream LSR’s Router ID, and the Downstream Interface Address MUST be set to the index assigned by the upstream LSR to the interface.

If an LSR does not know the IP address of its neighbor, then it MUST set the Address Type to either IPv4 Unnumbered or IPv6 Unnumbered. For IPv4, it must set the Downstream IP Address to 127.0.0.1; for IPv6 the address is set to 0::1. In both cases, the interface index MUST be set to 0. If an LSR receives an Echo Request packet with either of these addresses in the Downstream IP Address field, this indicates that it MUST bypass interface verification but continue with label validation.

If the originator of an Echo Request packet wishes to obtain Downstream Mapping information but does not know the expected label stack, then it SHOULD set the Address Type to either IPv4 Unnumbered or IPv6 Unnumbered. For IPv4, it MUST set the Downstream IP Address to 224.0.0.2; for IPv6 the address MUST be set to FF02::2. In both cases, the interface index MUST be set to 0. If an LSR receives an Echo Request packet with the all-routers multicast address, then this indicates that it MUST bypass both interface and label stack validation, but return Downstream Mapping TLVs using the information provided.

Multipath Type

The following Multipath Types are defined:

Key	Type	Multipath Information
0	no multipath	Empty (Multipath Length = 0)
2	IP address	IP addresses
4	IP address range	low/high address pairs
8	Bit-masked IP address set	IP address prefix and bit mask
9	Bit-masked label set	Label prefix and bit mask

Type 0 indicates that all packets will be forwarded out this one interface.

Types 2, 4, 8, and 9 specify that the supplied Multipath Information will serve to exercise this path.

Depth Limit

The Depth Limit is applicable only to a label stack and is the maximum number of labels considered in the hash; this SHOULD be set to zero if unspecified or unlimited.

Multipath Length

The length in octets of the Multipath Information.

Multipath Information

Address or label values encoded according to the Multipath Type. See the next section below for encoding details.

Downstream Label(s)

The set of labels in the label stack as it would have appeared if this router were forwarding the packet through this interface. Any Implicit Null labels are explicitly included. Labels are treated as numbers, i.e., they are right justified in the field.

A Downstream Label is 24 bits, in the same format as an MPLS label minus the TTL field, i.e., the MSBit of the label is bit 0, the LSBit is bit 19, the EXP bits are bits 20-22, and bit 23 is the S bit. The replying router SHOULD fill in the EXP and S bits; the LSR receiving the echo reply MAY choose to ignore these bits.

Protocol

The Protocol is taken from the following table:

Protocol #	Signaling Protocol
0	Unknown
1	Static
2	BGP
3	LDP
4	RSVP-TE

3.3.1. Multipath Information Encoding

The Multipath Information encodes labels or addresses that will exercise this path. The Multipath Information depends on the Multipath Type. The contents of the field are shown in the table above. IPv4 addresses are drawn from the range 127/8; IPv6 addresses are drawn from the range 0:0:0:0:0:FFFF:127/104. Labels are treated as numbers, i.e., they are right justified in the field. For Type 4, ranges indicated by Address pairs MUST NOT overlap and MUST be in ascending sequence.

Type 8 allows a more dense encoding of IP addresses. The IP prefix is formatted as a base IP address with the non-prefix low-order bits set to zero. The maximum prefix length is 27. Following the prefix is a mask of length 2^(32-prefix length) bits for IPv4 and 2^(128-prefix length) bits for IPv6. Each bit set to 1 represents a valid address. The address is the base IPv4 address plus the position of the bit in the mask where the bits are numbered left to right beginning with zero. For example, the IPv4 addresses 127.2.1.0, 127.2.1.5-127.2.1.15, and 127.2.1.20-127.2.1.29 would be encoded as follows:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0 1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Those same addresses embedded in IPv6 would be encoded as follows:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |0 1 1 1 1 1 1 1 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 1 1 1 1 1 1 1 1 1 1 0 0|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type 9 allows a more dense encoding of labels. The label prefix is formatted as a base label value with the non-prefix low-order bits set to zero. The maximum prefix (including leading zeros due to encoding) length is 27. Following the prefix is a mask of length 2^(32-prefix length) bits. Each bit set to one represents a valid label. The label is the base label plus the position of the bit in the mask where the bits are numbered left to right beginning with zero. Label values of all the odd numbers between 1152 and 1279 would be encoded as follows:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 +-
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |0 0
   0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 0 0 0 0 0 0| +-+-+-
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |0 1 0 1
   0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1| +-+-+-+-+-
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |0 1 0 1 0 1
   0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1| +-+-+-+-+-+-+-
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |0 1 0 1 0 1 0 1
   0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1| +-+-+-+-+-+-+-+-+-
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ |0 1 0 1 0 1 0 1 0 1
   0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1| +-+-+-+-+-+-+-+-+-
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

If the received Multipath Information is non-null, the labels and IP addresses MUST be picked from the set provided. If none of these labels or addresses map to a particular downstream interface, then for that interface, the type MUST be set to 0. If the received Multipath Information is null (i.e., Multipath Length = 0, or for Types 8 and 9, a mask of all zeros), the type MUST be set to 0.

For example, suppose LSR X at hop 10 has two downstream LSRs, Y and Z, for the FEC in question. The received X could return Multipath Type 4, with low/high IP addresses of 127.1.1.1->127.1.1.255 for downstream LSR Y and 127.2.1.1->127.2.1.255 for downstream LSR Z. The head end reflects this information to LSR Y. Y, which has three downstream LSRs, U, V, and W, computes that 127.1.1.1->127.1.1.127 would go to U and 127.1.1.128-> 127.1.1.255 would go to V. Y would then respond with 3 Downstream Mappings: to U, with Multipath Type 4 (127.1.1.1->127.1.1.127); to V, with Multipath Type 4 (127.1.1.127->127.1.1.255); and to W, with Multipath Type 0.

Note that computing Multipath Information may impose a significant processing burden on the receiver. A receiver MAY thus choose to process a subset of the received prefixes. The sender, on receiving a reply to a Downstream Mapping with partial information, SHOULD assume that the prefixes missing in the reply were skipped by the receiver, and MAY re-request information about them in a new echo request.

3.3.2. Downstream Router and Interface

The notion of "downstream router" and "downstream interface" should be explained. Consider an LSR X. If a packet that was originated with TTL n>1 arrived with outermost label L and TTL=1 at LSR X, X must be able to compute which LSRs could receive the packet if it was originated with TTL=n+1, over which interface the request would arrive and what label stack those LSRs would see. (It is outside the scope of this document to specify how this computation is done.) The set of these LSRs/interfaces consists of the downstream routers/interfaces (and their corresponding labels) for X with respect to L. Each pair of downstream router and interface requires a separate Downstream Mapping to be added to the reply.

The case where X is the LSR originating the echo request is a special case. X needs to figure out what LSRs would receive the MPLS echo request for a given FEC Stack that X originates with TTL=1.

The set of downstream routers at X may be alternative paths (see the discussion below on ECMP) or simultaneous paths (e.g., for MPLS multicast). In the former case, the Multipath Information is used as a hint to the sender as to how it may influence the choice of these alternatives.

3.4. Pad TLV

The value part of the Pad TLV contains a variable number (>= 1) of octets. The first octet takes values from the following table; all the other octets (if any) are ignored. The receiver SHOULD verify that the TLV is received in its entirety, but otherwise ignores the contents of this TLV, apart from the first octet.

Value	Meaning
1	Drop Pad TLV from reply
2	Copy Pad TLV to reply
3-255	Reserved for future use

3.5. Vendor Enterprise Number

SMI Private Enterprise Numbers are maintained by IANA. The Length is always 4; the value is the SMI Private Enterprise code, in network octet order, of the vendor with a Vendor Private extension to any of the fields in the fixed part of the message, in which case this TLV MUST be present. If none of the fields in the fixed part of the message have Vendor Private extensions, inclusion of this TLV is OPTIONAL. Vendor Private ranges for Message Types, Reply Modes, and Return Codes have been defined. When any of these are used, the Vendor Enterprise Number TLV MUST be included in the message.

3.6. Interface and Label Stack

The Interface and Label Stack TLV MAY be included in a reply message to report the interface on which the request message was received and the label stack that was on the packet when it was received. Only one such object may appear. The purpose of the object is to allow the upstream router to obtain the exact interface and label stack information as it appears at the replying LSR.

The Length is K + 4*N octets; N is the number of labels in the label stack. Values for K are found in the description of Address Type below. The Value field of a Downstream Mapping has the following format:

```
    0                   1                   2                   3
    0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Address Type  |             Must Be Zero                      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                   IP Address (4 or 16 octets)                 |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                   Interface (4 or 16 octets)                  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   .                                                               .
   .                                                               .
   .                          Label Stack                          .
   .                                                               .
   .                                                               .
   .                                                               .
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Address Type

The Address Type indicates if the interface is numbered or unnumbered. It also determines the length of the IP Address and Interface fields. The resulting total for the initial part of the TLV is listed in the table below as "K Octets". The Address Type is set to one of the following values:

Type #	Address Type	K Octets
1	IPv4 Numbered	12
2	IPv4 Unnumbered	12
3	IPv6 Numbered	36
4	IPv6 Unnumbered	24

IP Address and Interface

IPv4 addresses and interface indices are encoded in 4 octets; IPv6 addresses are encoded in 16 octets.

If the interface upon which the echo request message was received is numbered, then the Address Type MUST be set to IPv4 or IPv6, the IP Address MUST be set to either the LSR’s Router ID or the interface address, and the Interface MUST be set to the interface address.

If the interface is unnumbered, the Address Type MUST be either IPv4 Unnumbered or IPv6 Unnumbered, the IP Address MUST be the LSR’s Router ID, and the Interface MUST be set to the index assigned to the interface.

Label Stack

The label stack of the received echo request message. If any TTL values have been changed by this router, they SHOULD be restored.

3.7. Errored TLVs

The following TLV is a TLV that MAY be included in an echo reply to inform the sender of an echo request of mandatory TLVs either not supported by an implementation or parsed and found to be in error.

The Value field contains the TLVs that were not understood, encoded as sub-TLVs.

```
       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |             Type = 9          |            Length             |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      |                             Value                             |
      .                                                               .
      .                                                               .
      .                                                               .
      |                                                               |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

3.8. Reply TOS Byte TLV

This TLV MAY be used by the originator of the echo request to request that an echo reply be sent with the IP header TOS byte set to the value specified in the TLV. This TLV has a length of 4 with the following value field.

```
       0                   1                   2                   3
       0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      | Reply-TOS Byte|                 Must Be Zero                  |
      +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

4. Theory of Operation

An MPLS echo request is used to test a particular LSP. The LSP to be tested is identified by the "FEC Stack"; for example, if the LSP was set up via LDP, and is to an egress IP address of 10.1.1.1, the FEC Stack contains a single element, namely, an LDP IPv4 prefix sub-TLV with value 10.1.1.1/32. If the LSP being tested is an RSVP LSP, the FEC Stack consists of a single element that captures the RSVP Session and Sender Template that uniquely identifies the LSP.

FEC Stacks can be more complex. For example, one may wish to test a VPN IPv4 prefix of 10.1/8 that is tunneled over an LDP LSP with egress 10.10.1.1. The FEC Stack would then contain two sub-TLVs, the bottom being a VPN IPv4 prefix, and the top being an LDP IPv4 prefix. If the underlying (LDP) tunnel were not known, or was considered irrelevant, the FEC Stack could be a single element with just the VPN IPv4 sub-TLV.

When an MPLS echo request is received, the receiver is expected to verify that the control plane and data plane are both healthy (for the FEC Stack being pinged) and that the two planes are in sync. The procedures for this are in section 4.4 below.

4.1. Dealing with Equal-Cost Multi-Path (ECMP)

LSPs need not be simple point-to-point tunnels. Frequently, a single LSP may originate at several ingresses, and terminate at several egresses; this is very common with LDP LSPs. LSPs for a given FEC may also have multiple "next hops" at transit LSRs. At an ingress, there may also be several different LSPs to choose from to get to the desired endpoint. Finally, LSPs may have backup paths, detour paths, and other alternative paths to take should the primary LSP go down.

To deal with the last two first: it is assumed that the LSR sourcing MPLS echo requests can force the echo request into any desired LSP, so choosing among multiple LSPs at the ingress is not an issue. The problem of probing the various flavors of backup paths that will typically not be used for forwarding data unless the primary LSP is down will not be addressed here.

Since the actual LSP and path that a given packet may take may not be known a priori, it is useful if MPLS echo requests can exercise all possible paths. This, although desirable, may not be practical, because the algorithms that a given LSR uses to distribute packets over alternative paths may be proprietary.

To achieve some degree of coverage of alternate paths, there is a certain latitude in choosing the destination IP address and source

UDP port for an MPLS echo request. This is clearly not sufficient; in the case of traceroute, more latitude is offered by means of the Multipath Information of the Downstream Mapping TLV. This is used as follows. An ingress LSR periodically sends an MPLS traceroute message to determine whether there are multipaths for a given LSP. If so, each hop will provide some information how each of its downstream paths can be exercised. The ingress can then send MPLS echo requests that exercise these paths. If several transit LSRs have ECMP, the ingress may attempt to compose these to exercise all possible paths. However, full coverage may not be possible.

4.2. Testing LSPs That Are Used to Carry MPLS Payloads

To detect certain LSP breakages, it may be necessary to encapsulate an MPLS echo request packet with at least one additional label when testing LSPs that are used to carry MPLS payloads (such as LSPs used to carry L2VPN and L3VPN traffic. For example, when testing LDP or RSVP-TE LSPs, just sending an MPLS echo request packet may not detect instances where the router immediately upstream of the destination of the LSP ping may forward the MPLS echo request successfully over an interface not configured to carry MPLS payloads because of the use of penultimate hop popping. Since the receiving router has no means to differentiate whether the IP packet was sent unlabeled or implicitly labeled, the addition of labels shimmed above the MPLS echo request (using the Nil FEC) will prevent a router from forwarding such a packet out unlabeled interfaces.

4.3. Sending an MPLS Echo Request

An MPLS echo request is a UDP packet. The IP header is set as follows: the source IP address is a routable address of the sender; the destination IP address is a (randomly chosen) IPv4 address from the range 127/8 or IPv6 address from the range 0:0:0:0:0:FFFF:127/104. The IP TTL is set to 1. The source UDP port is chosen by the sender; the destination UDP port is set to 3503 (assigned by IANA for MPLS echo requests). The Router Alert option MUST be set in the IP header.

An MPLS echo request is sent with a label stack corresponding to the FEC Stack being tested. Note that further labels could be applied if, for example, the normal route to the topmost FEC in the stack is via a Traffic Engineered Tunnel [RSVP-TE]. If all of the FECs in the stack correspond to Implicit Null labels, the MPLS echo request is considered unlabeled even if further labels will be applied in sending the packet.

If the echo request is labeled, one MAY (depending on what is being pinged) set the TTL of the innermost label to 1, to prevent the ping request going farther than it should. Examples of where this SHOULD be done include pinging a VPN IPv4 or IPv6 prefix, an L2 VPN endpoint or a pseudowire. Preventing the ping request from going too far can also be accomplished by inserting a Router Alert label above this label; however, this may lead to the undesired side effect that MPLS echo requests take a different data path than actual data. For more information on how these mechanisms can be used for pseudowire connectivity verification, see [VCCV].

In "ping" mode (end-to-end connectivity check), the TTL in the outermost label is set to 255. In "traceroute" mode (fault isolation mode), the TTL is set successively to 1, 2, and so on.

The sender chooses a Sender’s Handle and a Sequence Number. When sending subsequent MPLS echo requests, the sender SHOULD increment the Sequence Number by 1. However, a sender MAY choose to send a group of echo requests with the same Sequence Number to improve the chance of arrival of at least one packet with that Sequence Number.

The TimeStamp Sent is set to the time-of-day (in seconds and microseconds) that the echo request is sent. The TimeStamp Received is set to zero.

An MPLS echo request MUST have an FEC Stack TLV. Also, the Reply Mode must be set to the desired reply mode; the Return Code and Subcode are set to zero. In the "traceroute" mode, the echo request SHOULD include a Downstream Mapping TLV.

4.4. Receiving an MPLS Echo Request

Sending an MPLS echo request to the control plane is triggered by one of the following packet processing exceptions: Router Alert option, IP TTL expiration, MPLS TTL expiration, MPLS Router Alert label, or the destination address in the 127/8 address range. The control plane further identifies it by UDP destination port 3503.

For reporting purposes the bottom of stack is considered to be stack-depth of 1. This is to establish an absolute reference for the case where the actual stack may have more labels than there are FECs in the Target FEC Stack.

Furthermore, in all the error codes listed in this document, a stack-depth of 0 means "no value specified". This allows compatibility with existing implementations that do not use the Return Subcode field.

An LSR X that receives an MPLS echo request then processes it as follows.

1. General packet sanity is verified. If the packet is not well-formed, LSR X SHOULD send an MPLS Echo Reply with the Return Code set to "Malformed echo request received" and the Subcode to zero. If there are any TLVs not marked as "Ignore" that LSR X does not understand, LSR X SHOULD send an MPLS "TLV not understood" (as appropriate), and the Subcode set to zero. In the latter case, the misunderstood TLVs (only) are included as sub-TLVs in an Errored TLVs TLV in the reply. The header fields Sender’s Handle, Sequence Number, and Timestamp Sent are not examined, but are included in the MPLS echo reply message.

The algorithm uses the following variables and identifiers:

Interface-I: the interface on which the MPLS echo request was received.

Stack-R: the label stack on the packet as it was received.

Stack-D: the label stack carried in the Downstream Mapping TLV (not always present)

Label-L: the label from the actual stack currently being examined. Requires no initialization.

Label-stack-depth: the depth of label being verified. Initialized to the number of labels in the received label stack S.

FEC-stack-depth: depth of the FEC in the Target FEC Stack that should be used to verify the current actual label. Requires no initialization.

Best-return-code: contains the return code for the echo reply packet as currently best known. As algorithm progresses, this code may change depending on the results of further checks that it performs.

Best-rtn-subcode: similar to Best-return-code, but for the Echo Reply Subcode.

FEC-status: result value returned by the FEC Checking algorithm described in section 4.4.1.

```
/* Save receive context information */
```

```
2. If the echo request is good, LSR X stores the interface over
   which the echo was received in Interface-I, and the label stack
   with which it came in Stack-R.

/* The rest of the algorithm iterates over the labels in Stack-R,
   verifies validity of label values, reports associated label
   switching operations (for traceroute), verifies correspondence
   between the Stack-R and the Target FEC Stack description in the
   body of the echo request, and reports any errors. */

/* The algorithm iterates as follows. */

3. Label Validation:

   If Label-stack-depth is 0 {

   /* The LSR needs to report its being a tail-end for the LSP */

      Set FEC-stack-depth to 1, set Label-L to 3 (Implicit Null).
      Set Best-return-code to 3 ("Replying router is an egress for
      the FEC at stack depth"), set Best-rtn-subcode to the
      value of FEC-stack-depth (1) and go to step 5 (Egress
      Processing).
   }

   /* This step assumes there is always an entry for well-known
      label values */

   Set Label-L to the value extracted from Stack-R at depth
   Label-stack-depth.  Look up Label-L in the Incoming Label Map
   (ILM) to determine if the label has been allocated and an
   operation is associated with it.

   If there is no entry for L {

   /* Indicates a temporary or permanent label synchronization
      problem the LSR needs to report an error */

      Set Best-return-code to 11 ("No label entry at stack-depth")
      and Best-rtn-subcode to Label-stack-depth.  Go to step 7
      (Send Reply Packet).
   }

   Else {

      Retrieve the associated label operation from the
      corresponding NLFE and proceed to step 4 (Label Operation
      check).
   }

4. Label Operation Check

   If the label operation is "Pop and Continue Processing" {

   /* Includes Explicit Null and Router Alert label cases */

      Iterate to the next label by decrementing Label-stack-depth
      and loop back to step 3 (Label Validation).
   }

   If the label operation is "Swap or Pop and Switch based on Popped
      Label" {

      Set Best-return-code to 8 ("Label switched at stack-depth")
      and Best-rtn-subcode to Label-stack-depth to report transit
      switching.

      If a Downstream Mapping TLV is present in the received echo
      request {

         If the IP address in the TLV is 127.0.0.1 or 0::1 {
            Set Best-return-code to 6 ("Upstream Interface Index
            Unknown").  An Interface and Label Stack TLV SHOULD be
            included in the reply and filled with Interface-I and
            Stack-R.
         }

         Else {

            Verify that the IP address, interface address, and label
            stack in the Downstream Mapping TLV match Interface-I
            and Stack-R.  If there is a mismatch, set
            Best-return-code to 5, "Downstream Mapping Mismatch".
            An Interface and Label Stack TLV SHOULD be included in
            the reply and filled in based on Interface-I and
            Stack-R.  Go to step 7 (Send Reply Packet).
         }
      }


      For each available downstream ECMP path {

         Retrieve output interface from the NHLFE entry.

         /* Note: this return code is set even if Label-stack-depth
            is one */
         If the output interface is not MPLS enabled {

            Set Best-return-code to Return Code 9, "Label switched
            but no MPLS forwarding at stack-depth" and set
            Best-rtn-subcode to Label-stack-depth and goto
            Send_Reply_Packet.
         }

         If a Downstream Mapping TLV is present {

           A Downstream Mapping TLV SHOULD be included in the echo
           reply (see section 3.3) filled in with information about
           the current ECMP path.
         }
      }

      If no Downstream Mapping TLV is present, or the Downstream IP
         Address is set to the ALLROUTERS multicast address,
            go to step 7 (Send Reply Packet).

      If the "Validate FEC Stack" flag is not set and the LSR is not
      configured to perform FEC checking by default, go to step 7
      (Send Reply Packet).

   /* Validate the Target FEC Stack in the received echo request.

      First determine FEC-stack-depth from the Downstream Mapping
      TLV.  This is done by walking through Stack-D (the Downstream
      labels) from the bottom, decrementing the number of labels
      for each non-Implicit Null label, while incrementing
      FEC-stack-depth for each label.  If the Downstream Mapping TLV
      contains one or more Implicit Null labels, FEC-stack-depth
      may be greater than Label-stack-depth.  To be consistent with
      the above stack-depths, the bottom is considered to entry 1.
      */

      Set FEC-stack-depth to 0.  Set i to Label-stack-depth.

      While (i > 0 ) do {
         ++FEC-stack-depth.
         if Stack-D[FEC-stack-depth] != 3 (Implicit Null)
            --i.
      }

      If the number of labels in the FEC stack is greater
         than or equal to FEC-stack-depth {
            Perform the FEC Checking procedure (see subsection 4.4.1
            below).

            If FEC-status is 2, set Best-return-code to 10 ("Mapping
            for this FEC is not the given label at stack-depth").

            If the return code is 1, set Best-return-code to
            FEC-return-code and Best-rtn-subcode to FEC-stack-depth.
         }

         Go to step 7 (Send Reply Packet).
      }

   5. Egress Processing:

      /* These steps are performed by the LSR that identified itself
         as the tail-end LSR for an LSP. */

      If received echo request contains no Downstream Mapping TLV, or
         the Downstream IP Address is set to 127.0.0.1 or 0::1
            go to step 6 (Egress FEC Validation).

      Verify that the IP address, interface address, and label stack in
      the Downstream Mapping TLV match Interface-I and Stack-R.  If
      not, set Best-return-code to 5, "Downstream Mapping
      Mis-match".  A Received Interface and Label Stack TLV SHOULD be
      created for the echo response packet.  Go to step 7 (Send Reply
      Packet).

   6. Egress FEC Validation:

      /* This is a loop for all entries in the Target FEC Stack
         starting with FEC-stack-depth. */

      Perform FEC checking by following the algorithm described in
      subsection 4.4.1 for Label-L and the FEC at FEC-stack-depth.

      Set Best-return-code to FEC-code and Best-rtn-subcode to the
      value in FEC-stack-depth.

      If FEC-status (the result of the check) is 1,
         go to step 7 (Send Reply Packet).

      /* Iterate to the next FEC entry */

      ++FEC-stack-depth.
```

```
    If FEC-stack-depth > the number of FECs in the FEC-stack,
       go to step 7 (Send Reply Packet).

    If FEC-status is 0 {
       ++Label-stack-depth.
       If Label-stack-depth > the number of labels in Stack-R,
          Go to step 7 (Send Reply Packet).

       Label-L = extracted label from Stack-R at depth
          Label-stack-depth.
       Loop back to step 6 (Egress FEC Validation).
    }
```

7. Send Reply Packet:

   Send an MPLS echo reply with a Return Code of Best-return-code, and a Return Subcode of Best-rtn-subcode. Include any TLVs created during the above process. The procedures for sending the echo reply are found in subsection 4.4.1.

4.4.1. FEC Validation

/* This subsection describes validation of an FEC entry within the Target FEC Stack and accepts an FEC, Label-L, and Interface-I. The algorithm performs the following steps. */

1. Two return values, FEC-status and FEC-return-code, are initialized to 0.

2. If the FEC is the Nil FEC {

```
       If Label-L is either Explicit_Null or Router_Alert, return.

       Else {
          Set FEC-return-code to 10 ("Mapping for this FEC is not
          the given label at stack-depth").
          Set FEC-status to 1
          Return.
       }
    }
```

3. Check the FEC label mapping that describes how traffic received on the LSP is further switched or which application it is associated with. If no mapping exists, set FEC-return-code to Return 4, "Replying router has no mapping for the FEC at stack-depth". Set FEC-status to 1. Return.

4. If the label mapping for FEC is Implicit Null, set FEC-status to 2 and proceed to step 5. Otherwise, if the label mapping for FEC is

Label-L, proceed to step 5. Otherwise, set FEC-return-code to 10 ("Mapping for this FEC is not the given label at stack-depth"), set FEC-status to 1, and return.

5. This is a protocol check. Check what protocol would be used to advertise FEC. If it can be determined that no protocol associated with Interface-I would have advertised an FEC of that FEC-Type, set FEC-return-code to 12 ("Protocol not associated with interface at FEC stack-depth"). Set FEC-status to 1.

6. Return.

4.5. Sending an MPLS Echo Reply

An MPLS echo reply is a UDP packet. It MUST ONLY be sent in response to an MPLS echo request. The source IP address is a routable address of the replier; the source port is the well-known UDP port for LSP ping. The destination IP address and UDP port are copied from the source IP address and UDP port of the echo request. The IP TTL is set to 255. If the Reply Mode in the echo request is "Reply via an IPv4 UDP packet with Router Alert", then the IP header MUST contain the Router Alert IP option. If the reply is sent over an LSP, the topmost label MUST in this case be the Router Alert label (1) (see [LABEL-STACK]).

The format of the echo reply is the same as the echo request. The Sender’s Handle, the Sequence Number, and TimeStamp Sent are copied from the echo request; the TimeStamp Received is set to the time-of-day that the echo request is received (note that this information is most useful if the time-of-day clocks on the requester and the replier are synchronized). The FEC Stack TLV from the echo request MAY be copied to the reply.

The replier MUST fill in the Return Code and Subcode, as determined in the previous subsection.

If the echo request contains a Pad TLV, the replier MUST interpret the first octet for instructions regarding how to reply.

If the replying router is the destination of the FEC, then Downstream Mapping TLVs SHOULD NOT be included in the echo reply.

If the echo request contains a Downstream Mapping TLV, and the replying router is not the destination of the FEC, the replier SHOULD compute its downstream routers and corresponding labels for the incoming label, and add Downstream Mapping TLVs for each one to the echo reply it sends back.

If the Downstream Mapping TLV contains Multipath Information requiring more processing than the receiving router is willing to perform, the responding router MAY choose to respond with only a subset of multipaths contained in the echo request Downstream Mapping. (Note: The originator of the echo request MAY send another echo request with the Multipath Information that was not included in the reply.)

Except in the case of Reply Mode 4, "Reply via application level control channel", echo replies are always sent in the context of the IP/MPLS network.

4.6. Receiving an MPLS Echo Reply

An LSR X should only receive an MPLS echo reply in response to an MPLS echo request that it sent. Thus, on receipt of an MPLS echo reply, X should parse the packet to ensure that it is well-formed, then attempt to match up the echo reply with an echo request that it had previously sent, using the destination UDP port and the Sender’s Handle. If no match is found, then X jettisons the echo reply; otherwise, it checks the Sequence Number to see if it matches.

If the echo reply contains Downstream Mappings, and X wishes to traceroute further, it SHOULD copy the Downstream Mapping(s) into its next echo request(s) (with TTL incremented by one).

4.7. Issue with VPN IPv4 and IPv6 Prefixes

Typically, an LSP ping for a VPN IPv4 prefix or VPN IPv6 prefix is sent with a label stack of depth greater than 1, with the innermost label having a TTL of 1. This is to terminate the ping at the egress PE, before it gets sent to the customer device. However, under certain circumstances, the label stack can shrink to a single label before the ping hits the egress PE; this will result in the ping terminating prematurely. One such scenario is a multi-AS Carrier’s Carrier VPN.

To get around this problem, one approach is for the LSR that receives such a ping to realize that the ping terminated prematurely, and send back error code 13. In that case, the initiating LSR can retry the ping after incrementing the TTL on the VPN label. In this fashion, the ingress LSR will sequentially try TTL values until it finds one that allows the VPN ping to reach the egress PE.

4.8. Non-compliant Routers

If the egress for the FEC Stack being pinged does not support MPLS ping, then no reply will be sent, resulting in possible "false negatives". If in "traceroute" mode, a transit LSR does not support LSP ping, then no reply will be forthcoming from that LSR for some TTL, say, n. The LSR originating the echo request SHOULD try sending the echo request with TTL=n+1, n+2, ..., n+k to probe LSRs further down the path. In such a case, the echo request for TTL > n SHOULD be sent with Downstream Mapping TLV "Downstream IP Address" field set to the ALLROUTERs multicast address until a reply is received with a Downstream Mapping TLV. The label stack MAY be omitted from the Downstream Mapping TLV. Furthermore, the "Validate FEC Stack" flag SHOULD NOT be set until an echo reply packet with a Downstream Mapping TLV is received.

5. References

5.1. Normative References

[BGP] Rekhter, Y., Li, T., and S. Hares, "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, January 2006.

[IANA] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[KEYWORDS] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[LABEL-STACK] Rosen, E., Tappan, D., Fedorkow, G., Rekhter, Y., Farinacci, D., Li, T., and A. Conta, "MPLS Label Stack Encoding", RFC 3032, January 2001.

[NTP] Mills, D., "Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI", RFC 2030, October 1996.

[RFC1122] Braden, R., "Requirements for Internet Hosts - Communication Layers", STD 3, RFC 1122, October 1989.

[RFC1812] Baker, F., "Requirements for IP Version 4 Routers", RFC 1812, June 1995.

[RFC4026] Andersson, L. and T. Madsen, "Provider Provisioned Virtual Private Network (VPN) Terminology", RFC 4026, March 2005.

5.2. Informative References

[BGP-LABEL] Rekhter, Y. and E. Rosen, "Carrying Label Information in BGP-4", RFC 3107, May 2001.

[ICMP] Postel, J., "Internet Control Message Protocol", STD 5, RFC 792, September 1981.

[LDP] Andersson, L., Doolan, P., Feldman, N., Fredette, A., and B. Thomas, "LDP Specification", RFC 3036, January 2001.

[PW-CONTROL] Martini, L., El-Aawar, N., Heron, G., Rosen, E., Tappan, D., and T. Smith, "Pseudowire Setup and Maintenance using the Label Distribution Protocol", Work in Progress.

[RFC4365] Rosen, E., "Applicability Statement for BGP/MPLS IP Virtual Private Networks (VPNs)", RFC 4365, February 2006.

[RSVP-TE] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, December 2001.

[VCCV] Nadeau, T. and R. Aggarwal, "Pseudo Wire Virtual Circuit Connectivity Verification (VCCV), Work in Progress, August 2005.

[VPLS-BGP] Kompella, K. and Y. Rekhter, "Virtual Private LAN Service", Work in Progress.

6. Security Considerations

Overall, the security needs for LSP ping are similar to those of ICMP ping.

There are at least three approaches to attacking LSRs using the mechanisms defined here. One is a Denial-of-Service attack, by sending MPLS echo requests/replies to LSRs and thereby increasing their workload. The second is obfuscating the state of the MPLS data plane liveness by spoofing, hijacking, replaying, or otherwise tampering with MPLS echo requests and replies. The third is an unauthorized source using an LSP ping to obtain information about the network.

To avoid potential Denial-of-Service attacks, it is RECOMMENDED that implementations regulate the LSP ping traffic going to the control plane. A rate limiter SHOULD be applied to the well-known UDP port defined below.

Unsophisticated replay and spoofing attacks involving faking or replaying MPLS echo reply messages are unlikely to be effective. These replies would have to match the Sender’s Handle and Sequence Number of an outstanding MPLS echo request message. A non-matching replay would be discarded as the sequence has moved on, thus a spoof has only a small window of opportunity. However, to provide a stronger defense, an implementation MAY also validate the TimeStamp Sent by requiring and exact match on this field.

To protect against unauthorized sources using MPLS echo request messages to obtain network information, it is RECOMMENDED that implementations provide a means of checking the source addresses of MPLS echo request messages against an access list before accepting the message.

It is not clear how to prevent hijacking (non-delivery) of echo requests or replies; however, if these messages are indeed hijacked, LSP ping will report that the data plane is not working as it should.

It does not seem vital (at this point) to secure the data carried in MPLS echo requests and replies, although knowledge of the state of the MPLS data plane may be considered confidential by some. Implementations SHOULD, however, provide a means of filtering the addresses to which echo reply messages may be sent.

Although this document makes special use of 127/8 address, these are used only in conjunction with the UDP port 3503. Furthermore, these packets are only processed by routers. All other hosts MUST treat all packets with a destination address in the range 127/8 in accordance to RFC 1122. Any packet received by a router with a destination address in the range 127/8 without a destination UDP port of 3503 MUST be treated in accordance to RFC 1812. In particular, the default behavior is to treat packets destined to a 127/8 address as "martians".

7. IANA Considerations

The TCP and UDP port number 3503 has been allocated by IANA for LSP echo requests and replies.

The following sections detail the new name spaces to be managed by IANA. For each of these name spaces, the space is divided into assignment ranges; the following terms are used in describing the procedures by which IANA allocates values: "Standards Action" (as defined in [IANA]), "Specification Required", and "Vendor Private Use".

Values from "Specification Required" ranges MUST be registered with IANA. The request MUST be made via an Experimental RFC that describes the format and procedures for using the code point; the actual assignment is made during the IANA actions for the RFC.

Values from "Vendor Private" ranges MUST NOT be registered with IANA; however, the message MUST contain an enterprise code as registered with the IANA SMI Private Network Management Private Enterprise Numbers. For each name space that has a Vendor Private range, it must be specified where exactly the SMI Private Enterprise Number resides; see below for examples. In this way, several enterprises (vendors) can use the same code point without fear of collision.

7.1. Message Types, Reply Modes, Return Codes

The IANA has created and will maintain registries for Message Types, Reply Modes, and Return Codes. Each of these can take values in the range 0-255. Assignments in the range 0-191 are via Standards Action; assignments in the range 192-251 are made via "Specification Required"; values in the range 252-255 are for Vendor Private Use, and MUST NOT be allocated.

If any of these fields fall in the Vendor Private range, a top-level Vendor Enterprise Number TLV MUST be present in the message.

Message Types defined in this document are the following:

Value	Meaning
1	MPLS echo request
2	MPLS echo reply

Reply Modes defined in this document are the following:

Value	Meaning
1	Do not reply
2	Reply via an IPv4/IPv6 UDP packet
3	Reply via an IPv4/IPv6 UDP packet with Router Alert
4	Reply via application level control channel

Return Codes defined in this document are listed in section 3.1.

7.2. TLVs

The IANA has created and will maintain a registry for the Type field of top-level TLVs as well as for any associated sub-TLVs. Note the meaning of a sub-TLV is scoped by the TLV. The number spaces for the sub-TLVs of various TLVs are independent.

The valid range for TLVs and sub-TLVs is 0-65535. Assignments in the range 0-16383 and 32768-49161 are made via Standards Action as defined in [IANA]; assignments in the range 16384-31743 and 49162-64511 are made via "Specification Required" as defined above; values in the range 31744-32767 and 64512-65535 are for Vendor Private Use, and MUST NOT be allocated.

If a TLV or sub-TLV has a Type that falls in the range for Vendor Private Use, the Length MUST be at least 4, and the first four octets MUST be that vendor’s SMI Private Enterprise Number, in network octet order. The rest of the Value field is private to the vendor.

TLVs and sub-TLVs defined in this document are the following:

Type	Sub-Type	Value Field
1		Target FEC Stack
	1	LDP IPv4 prefix
	2	LDP IPv6 prefix
	3	RSVP IPv4 LSP
	4	RSVP IPv6 LSP
	5	Not Assigned
	6	VPN IPv4 prefix
	7	VPN IPv6 prefix
	8	L2 VPN endpoint
	9	"FEC 128" Pseudowire (Deprecated)
	10	"FEC 128" Pseudowire
	11	"FEC 129" Pseudowire
	12	BGP labeled IPv4 prefix
	13	BGP labeled IPv6 prefix
	14	Generic IPv4 prefix
	15	Generic IPv6 prefix
	16	Nil FEC
2		Downstream Mapping
3		Pad
4		Not Assigned
5		Vendor Enterprise Number
6		Not Assigned
7		Interface and Label Stack
8		Not Assigned
9		Errored TLVs
	Any value	The TLV not understood
10		Reply TOS Byte

8. Acknowledgements

This document is the outcome of many discussions among many people, including Manoj Leelanivas, Paul Traina, Yakov Rekhter, Der-Hwa Gan, Brook Bailey, Eric Rosen, Ina Minei, Shivani Aggarwal, and Vanson Lim.

The description of the Multipath Information sub-field of the Downstream Mapping TLV was adapted from text suggested by Curtis Villamizar.

Authors’ Addresses

Kireeti Kompella
Juniper Networks
1194 N.Mathilda Ave
Sunnyvale, CA 94089

EMail: kireeti@juniper.net

George Swallow
Cisco Systems
1414 Massachusetts Ave,
Boxborough, MA 01719

Phone: +1 978 936 1398
EMail: swallow@cisco.com

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).